DBMS PROJECT

E_COMMERCE MANAGEMENT SYSTEM

TEAM MEMBERS:

ABINAYASRI S. 23BCS001

AMIRTHAVARSHNI J. 23BCS002

ANUBARATHI M. 23BCS003

ANUGRAHAA V. 23BCS004

ANUKEERTHANA P. 23BCS005

GUIDED BY:

Mrs. J. GAYATHRI, Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

OCTOBER 2024

CASE STUDY:

E-commerce(online shopping and delivery management system).

BACKGROUND:

E-commerce, short for electronic commerce, refers to the buying and selling of goods or services over the internet. The rise of e-commerce shopping can be traced back to the late 1990s and early 2000s, but it gained significant momentum with the advancement of technology, faster internet speeds, and the increasing use of smartphones.

PROJECT SCOPE:

- The scope of e-commerce includes managing an online product catalog, processing customer orders, and handling payment transactions securely.
- Additionally, implementing robust backup and recovery strategies is essential for data protection.

CHALLENGES IN E-COMMERCE:

- **SECURITY CONCERNS:** With the rise of online transactions, there is an increased risk of fraud, data breaches, and cybersecurity threats.
- **COMPETITION:** The low barrier to entry in e-commerce has led to intense competition among online retailers, making it difficult for smaller businesses to stand out.

- **LOGISTICS AND DELIVERY:** Efficient order fulfillment and timely delivery remain critical factors for customer satisfaction. Managing logistics, especially for global customers, can be a challenge.

CONCLUSION:

- In conclusion, an e-commerce project for online shopping relies on a robust DBMS to manage product information, customer data, and transactions efficiently.

- Effective backup and recovery processes are essential for data protection and system reliability.

ER DIAGRAM

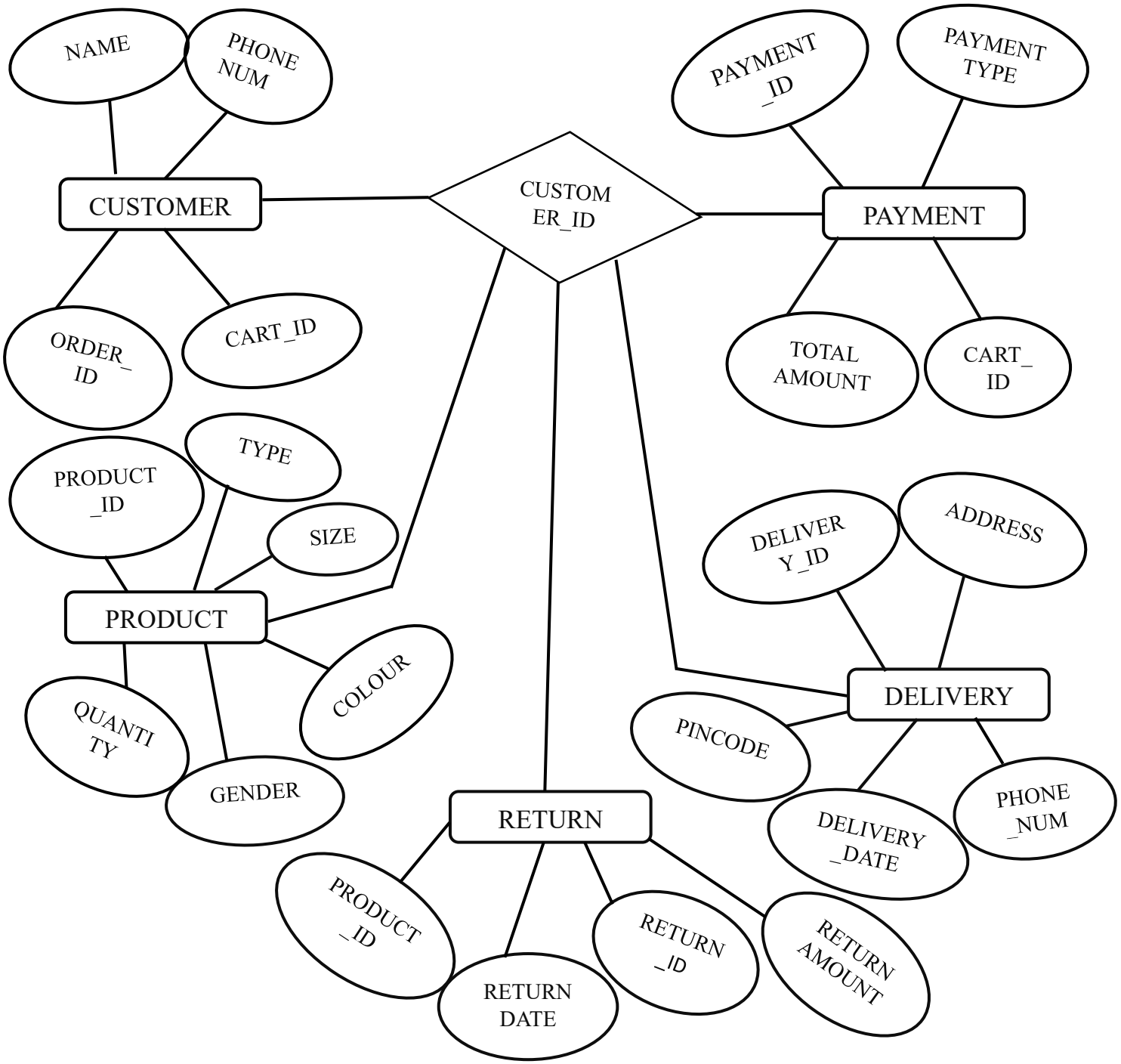
NAME
PHONE NUM
CUSTOMER
CUSTOM ER_ID
PAYMENT _ID
PAYMENT TYPE
PAYMENT
ORDER_ ID
CART_ID
TOTAL AMOUNT
CART_ ID
PRODUCT _ID
TYPE
SIZE
PRODUCT
DELIVER Y_ID
ADDRESS
DELIVERY
QUANTI TY
COLOUR
GENDER
PINCODE
PHONE _NUM
DELIVERY _DATE
RETURN
PRODUCT _ID
RETURN DATE
RETURN _ID
RETURN AMOUNT

TABLE NAME: CUSTOMER

COLOUMN NAME	DATA TYPE	INDEX	DESCRIPTION
CUSTOMER_ID	INT	PRIMARY KEY	Unique identifier for each customer.
CART_ID	INT		Unique cart_id for each customer.
NAME	VARCHAR		Customer's name.
PHONE	INT		Customer's mobile number.
ORDER_DATE	INT		Ordered date.

TABLE NAME: PRODUCT

COLOUMN NAME	DATA TYPE	INDEX	DESCRIPTION
PRODUCT_ID	INT	PRIMARY KEY	Unique identifier for each product.
CUSTOMER_ID	INT	FOREIGN KEY	Reference the customer_id from the customer table.
TYPE	VARCHAR		Indicates the type of our search.
COLOUR	VARCHAR		It indicates the colour for the product.
GENDER	VARCHAR		The gender helps to find the product much faster.
QUANTITY	INT		How many you want to buy.(1 or2,3).

TABLE NAME: PAYMENT

COLOUMN NAME	DATA TYPE	INDEX	DESCRIPTION
PAYMENT_ID	INT	PRIMARY KEY	Unique identifier for each payment method
CUSTOMER_ID	VARCHAR	FOREIGN KEY	Reference the customer_id from the customer table.

CART_ID	INT		Unique cart_id for each customer.
PAYMENT_TYPE	INT		Indicates the type or method of the payment.
TOTAL_AMOUNT	INT		Indicates the total amount of the order.

TABLE NAME: DELIVERY

COLOUMN NAME	**DATA TYPE**	**INDEX**	**DESCRIPTION**
DELIVERY_ID	INT	PRIMARY KEY	Unique identifier for each delivery.
CUSTOMER_ID	INT	FOREIGN KEY	Reference the customer_id from customer table.
ADDRESS	VARCHAR		Address in which the ordered product will be delivered.
PINCODE	INT		To represent the place where they live.
PHONE_NUM	INT		Customer's mobile number.
DELIVERY_DATE	INT		On which the product will be delivered

TABLE NAME: RETURN

COLOUMN NAME	**DATA TYPE**	**INDEX**	**DESCRIPTION**
RETURN_ID	INT	PRIMARY KEY	Unique identification for each return option.
CUSTOMER_ID	INT	FOREIGN KEY	It refers the customer_id from customer table.
PRODUCT_ID	INT	FOREIGN KEY	Refers to the product_id from the product table.
RETURN_DATE	INT		It represent the product return date.
RETURN_AMOUNT	INT		It represents the amount that the seller would return to you.

TABLE CREATEION AND INSERTION CODE

SQL> create table customer(customer_id number(10) primary key,name varchar2(30),phone_num number(10),order_date date,cart_id number(10));

Table created.

SQL> insert into customer values (&customer_id, '&name', &phone_num, '&order_date', &cart_id);

Enter value for customer_id: 101

Enter value for name: abinaya

Enter value for phone_num: 9080766775

Enter value for order_date: 1-jan-2023

Enter value for cart_id: 01

1 row created.

SQL> /

Enter value for customer_id: 102

Enter value for name: amirthavarshini

Enter value for phone_num: 7668889899

Enter value for order_date: 2-jan-2023

Enter value for cart_id: 02

1 row created.

SQL> /

Enter value for customer_id: 103

Enter value for name: anubarathi

Enter value for phone_num: 9080318180

Enter value for order_date: 3-jan-2023

Enter value for cart_id: 03

1 row created.

SQL> /

Enter value for customer_id: 104

Enter value for name: anugrahaa

Enter value for phone_num: 8223566789Enter value for order_date: 4-jan-2023

Enter value for cart_id: 04

1 row created.

SQL> /

Enter value for customer_id: 105

Enter value for name: anukeerthana

Enter value for phone_num: 7678887437

Enter value for order_date: 5-jan-2023

Enter value for cart_id: 05

1 row created.

SQL> /

Enter value for customer_id: 106

Enter value for name: lia

Enter value for phone_num: 8334679090

Enter value for order_date: 6-jan-2023

Enter value for cart_id: 06

1 row created.

SQL> /

Enter value for customer_id: 107

Enter value for name: raj

Enter value for phone_num: 7675435465

Enter value for order_date: 7-jan-2023

Enter value for cart_id: 07

1 row created.

SQL> /

Enter value for customer_id: 108

Enter value for name: vicky

Enter value for phone_num: 9614070476

Enter value for order_date: 8-jan-2023

Enter value for cart_id: 08

SQL> select*from customer;

CUSTOMER_ID	NAME	PHONE_NUM	ORDER_DAT	CART_ID
101	abinaya	9080766775	01-JAN-23	1
102	amirthavarshini	7668889899	02-JAN-23	2
103	anubarathi	9080318180	03-JAN-23	3
104	anugrahaa	8223566789	04-JAN-23	4
105	anukeerthana	7678887437	05-JAN-23	5
106	lia	8334679090	06-JAN-23	6
107	raj	7675435465	07-JAN-23	7
108	vicky	9614070476	08-JAN-23	8

SQL> create table cart(cart_id number primary key);

Table created.

SQL> insert into cart values(01);

1 row created.

insert into cart values(02);

1 row created.

SQL> insert into cart values(03);

1 row created.

SQL> insert into cart values(04);

1 row created.

SQL> insert into cart values(05);

1 row created.

SQL> insert into cart values(06);

1 row created.

SQL> insert into cart values(07);

1 row created.

SQL> insert into cart values(08);

1 row created.

SQL> create table payment(payment_id number(10) primary key,payment_type

varchar2(10),total_amount number(10),customer_id number references customer,cart_id

number(10)references cart);

Table created.

SQL> insert into payment ValueS (&payment_id, '&payment_type', &total_amount, &customer_id, &cart_id);

Enter value for payment_id: 201

Enter value for payment_type: upi

Enter value for total_amount: 1000

Enter value for customer_id: 101

Enter value for cart_id: 01

old 1: insert into payment

values(&payment_id,'&payment_type',&total_amount,&customer_id,&cart_id)

new 1: insert into payment values(201,'upi',1000,101,01)

1 row created.

SQL> /

Enter value for payment_id: 202

Enter value for payment_type: netbanking

Enter value for total_amount: 1500

Enter value for customer_id: 102

Enter value for cart_id: 02

1 row created.

SQL> /

Enter value for payment_id: 203

Enter value for payment_type: gpay

Enter value for total_amount: 2000

Enter value for customer_id: 103

Enter value for cart_id: 03

1 row created.

SQL> /

Enter value for payment_id: 204

Enter value for payment_type: online

Enter value for total_amount: 2500

Enter value for customer_id: 104

Enter value for cart_id: 04

1 row created.

SQL> /

Enter value for payment_id: 205

Enter value for payment_type: cod

Enter value for total_amount: 3000

Enter value for customer_id: 105

Enter value for cart_id: 05

1 row created.

SQL> /

Enter value for payment_id: 206

Enter value for payment_type: card

Enter value for total_amount: 3500

Enter value for customer_id: 106

Enter value for cart_id: 06

1 row created.

SQL> /

Enter value for payment_id: 207

Enter value for payment_type: online

Enter value for total_amount: 4000

Enter value for customer_id: 107

Enter value for cart_id: 07

1 row created.

SQL> /

Enter value for payment_id: 208

Enter value for payment_type: cod

Enter value for total_amount: 4500

Enter value for customer_id: 108

Enter value for cart_id: 08

1 row created.

SQL> select*from payment;

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
201	upi	1000	101	1
202	netbanking	1500	102	2
203	gpay	2000	103	3
204	online	2500	104	4
205	cod	3000	105	5
206	card	3500	106	6
207	online	4000	107	7
208	cod	4500	108	8

SQL> create table product(product_id number primary key,customer_id number references customer,type varchar2(20),colour varchar2(20),gender varchar2(10),quantity number(10));

Table created.

SQL> insert into product values (&product_id, &customer_id, '&type', '&colour', '&gender', &quantity);

Enter value for product_id: 301

Enter value for customer_id: 101

Enter value for type: western

Enter value for colour: orange

Enter value for gender: female

Enter value for quantity: 1

1 row created.

SQL> /

Enter value for product_id: 302

Enter value for customer_id: 102

Enter value for type: traditional

Enter value for colour: red

Enter value for gender: male

Enter value for quantity: 2

1 row created.

SQL> /

Enter value for product_id: 303

Enter value for customer_id: 103

Enter value for type: formal

Enter value for colour: green

Enter value for gender: male

Enter value for quantity: 1

1 row created.

SQL> /

Enter value for product_id: 304

Enter value for customer_id: 104

Enter value for type: ethnic

Enter value for colour: blue

Enter value for gender: female

Enter value for quantity: 2

1 row created.

SQL> /

Enter value for product_id: 305

Enter value for customer_id: 105

Enter value for type: casual

Enter value for colour: yellow

Enter value for gender: male

Enter value for quantity: 1

1 row created.

SQL> /

Enter value for product_id: 306

Enter value for customer_id: 106

Enter value for type: traditional

Enter value for colour: white

Enter value for gender: female

Enter value for quantity: 2

1 row created.

SQL> /

Enter value for product_id: 307

Enter value for customer_id: 107

Enter value for type: vintage

Enter value for colour: black

Enter value for gender: female

Enter value for quantity: 1

1 row created.

SQL> /

Enter value for product_id: 308

Enter value for customer_id: 108

Enter value for type: modern

Enter value for colour: pink

Enter value for gender: male

Enter value for quantity: 2

1 row created.

SQL> select*from product;

PRODUCT_ID	CUSTOMER_ID	TYPE	COLOUR	GENDER	QUANTITY
301	101	western	orange	female	1
302	102	traditional	red	male	2
303	103	formal	green	male	1
304	104	ethnic	blue	female	2
305	105	casual	yellow	male	1
306	106	traditional	white	female	2

307	107	vintage	black	female	1
308	108	modern	pink	male	2

SQL> create table delivery(delivery_id number primary key,customer_id number references customer,address varchar2(30),pincode number(6),phone_num number (10),delivery_date date);

Table created.

SQL> insert into delivery values (&delivery_id, &customer_id, '&address', &pincode, &phone_num, '&delivery_date');

Enter value for delivery_id: 401

Enter value for customer_id: 101

Enter value for address: coimbatore

Enter value for pincode: 653234

Enter value for phone_num: 9087658778

Enter value for delivery_date: 1-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 402

Enter value for customer_id: 102

Enter value for address: chennai

Enter value for pincode: 625987

Enter value for phone_num: 6547893426

Enter value for delivery_date: 2-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 403

Enter value for customer_id: 103

Enter value for address: madurai

Enter value for pincode: 654378

Enter value for phone_num: 9087656334

Enter value for delivery_date: 3-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 404

Enter value for customer_id: 104

Enter value for address: trichy

Enter value for pincode: 600987

Enter value for phone_num: 7865786543

Enter value for delivery_date: 4-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 405

Enter value for customer_id: 105

Enter value for address: kerala

Enter value for pincode: 656578

Enter value for phone_num: 765892345

Enter value for delivery_date: 5-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 406

Enter value for customer_id: 106

Enter value for address: pollachi

Enter value for pincode: 645789

Enter value for phone_num: 7685409876

Enter value for delivery_date: 6-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 407

Enter value for customer_id: 107

Enter value for address: erode

Enter value for pincode: 651342

Enter value for phone_num: 7689432567

Enter value for delivery_date: 7-feb-2023

1 row created.

SQL> /

Enter value for delivery_id: 408

Enter value for customer_id: 108

Enter value for address: andhrapradesh

Enter value for pincode: 675876

Enter value for phone_num: 8267568923

Enter value for delivery_date: 8-feb-2023

1 row created.

SQL> select*from delivery;

DELIVERY_ID	CUSTOMER_ID	ADDRESS	PINCODE	PHONE_NUM	DELIVERY_
401	101	coimbatore	653234	9087658778	01-FEB-23
402	102	chennai	625987	6547893426	02-FEB-23
403	103	madurai	654378	9087656334	03-FEB-23
404	104	trichy	600987	7865786543	04-FEB-23
405	105	kerala	656578	765892345	05-FEB-23
406	106	pollachi	645789	7685409876	06-FEB-23
407	107	erode	651342	7689432567	07-FEB-23
408	108	andhrapradesh	675876	8267568923	08-FEB-23

SQL> create table return(return_id number(10) primary key,customer_id number references customer,product_id number references product,return_date date,return_amount number(10));

Table created.

SQL> insert into return values(&return_id, &customer_id, &product_id, '&return_date', &return_amount);

Enter value for return_id: 501

Enter value for customer_id: 101

Enter value for product_id: 301

Enter value for return_date: 11-feb-2023

Enter value for return_amount: 1000

1 row created.

SQL> /

Enter value for return_id: 502

Enter value for customer_id: 102

Enter value for product_id: 302

Enter value for return_date: 12-feb-2023

Enter value for return_amount: 1500

1 row created.

SQL> /

Enter value for return_id: 503

Enter value for customer_id: 103

Enter value for product_id: 303

Enter value for return_date: 13-feb-2023

Enter value for return_amount: 2000

1 row created.

SQL> /

Enter value for return_id: 504

Enter value for customer_id: 104

Enter value for product_id: 304

Enter value for return_date: 14-feb-2023

Enter value for return_amount: 2500

1 row created.

SQL> /

Enter value for return_id: 505

Enter value for customer_id: 105

Enter value for product_id: 305

Enter value for return_date: 15-feb-2023

Enter value for return_amount: 3000

1 row created.

SQL> /

Enter value for return_id: 506

Enter value for customer_id: 106

Enter value for product_id: 306

Enter value for return_date: 16-feb-2023

Enter value for return_amount: 3500

1 row created.

SQL> /

Enter value for return_id: 507

Enter value for customer_id: 107

Enter value for product_id: 307

Enter value for return_date: 17-feb-2023

Enter value for return_amount: 4000

1 row created.

SQL> /

Enter value for return_id: 508

Enter value for customer_id: 108

Enter value for product_id: 308

Enter value for return_date: 18-feb-2023

Enter value for return_amount: 4500

1 row created.

SQL> select*from return;

RETURN_ID	CUSTOMER_ID	PRODUCT_ID	RETURN_DA	RETURN_AMOUNT
501	101	301	11-FEB-23	1000
502	102	302	12-FEB-23	1500
503	103	303	13-FEB-23	2000

504	104	304	14-FEB-23	2500
506	106	306	15-FEB-23	3000
507	107	307	16-FEB-23	3500
508	108	308	17-FEB-23	4000

DATA DEFINITION LANGUAGE

1. **Add a "age" column to customer table:**

   SQL> alter table customer add age varchar(5);

   Table altered.

2. **Drop the payment table:**

   SQL> drop table payment;

   Table dropped.

3. **Rename the customer table:**

   SQL> alter table customer rename to consumer;

   Table altered.

4. **Rename the "phone_num" column in customer(consumer):**

   SQL> alter table consumer rename column phone_num to phone_no;

   Table altered.

5. **Truncating a table remove all date but keeps the table structure:**

   SQL> truncate table delivery;

   Table truncated.

6. **Add a "delivery_time" column to delivery table:**

   SQL> alter table delivery add delivery_time int;

   Table altered.

7. **Renaming a table:**

   SQL> alter table consumer rename to customer;

   Table altered.

8. **Add a "monthly_emi" column to payment table:**

   SQL> alter table payment add monthly_emi number;

Table altered.

DATA MANIPULATION LANGUAGE

1. **Update the name of the customer in the customer table:**
   SQL>update customer information set name='lia' where customer_id=106;

   1 row updated.

2. **Increase the total amount for all customer in the payment table:**
   SQL>update payment set total_amount=total_amount*10;
   8 rows updated.

3. **Remove a specific payment type for a customer in the payment table:**
   SQL>delete from payment where payment_id=204 and payment_type='online';

   1 row deleted.

4. **Delete a customer's record for specific id:**
   SQL> delete from payment where customer_id=108;

   1 row deleted.

5. **Increase the total amount of customer with payment_id:**
   SQL>update payment set total_amount=total_amount*1.15 where payment_id=208;

   1 row updated.

6. **Remove all the records from the payment table where payment_type='card':**
   SQL> delete from payment where payment_type='card';

   1 row deleted.

7. **Change the phone_num of the customer in customer_if=108:**
   SQL>update customer set phone_num=9080454654 where customer_id=108;

   1 row updated.

8. **Change the payment_type of all payment methods in the payment table to 'cash':**
   SQL> update payment set payment_type='cash';
   6 rows updated.

9. **To select the name of the customer using customer_id:**
   SQL> select name from customer where customer_id=103;
   NAME
   ---
   anubarathi

10. **Calculate the total amount for the company:**
    SQL>select sum(total_amount) from payment;

SUM(TOTAL_AMOUNT)

166750

11. List customers with payment_id=201:

SQL>select*from payment where payment_id=201;

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
201	cash	10000	101	1

DATA INTERGRITY CONSTRAINTS

1. **Add a primary key constraint to an existing table:**
   SQL> ALTER TABLE payment ADD CONSTRAINT pk_payment PRIMARY KEY (PAYMENT_ID);
   Table altered.

2. **Add an Unique Constraint to an existing column:**
   SQL> ALTER TABLE payment ADD CONSTRAINT uq_customer_id UNIQUE (CUSTOMER_ID);
   Table altered.

3. **Add a foreign key constraint to an existing table:**
   SQL> ALTER TABLE payment ADD CONSTRAINT fk_payment_customer FOREIGN KEY (customer_id) REFERENCES customer(customer_id);

   Table altered.

4. **Create a table with a check constraint:**
   SQL> CREATE TABLE product (product_id NUMBER(10) PRIMARY KEY, product_name VARCHAR2(50),quantity NUMBER(10) CHECK (quantity >= 0), price NUMBER(10, 2) CHECK (price > 0));
   Table created.

5. **Add a check Constraint to an existing coloumn:**
   SQL> ALTER TABLE payment ADD CONSTRAINT chk_total_amount_non_negative CHECK (total_amount >= 0);
   Table altered.

6. **Add a default constraint to an existing column:**
   ALTER TABLE product MODIFY quantity DEFAULT 0;
   Table altered.

7. **Add a not null constraint to an existing column:**
   SQL> ALTER TABLE product MODIFY quantity NUMBER(10) NOT NULL;
   Table altered

TRANSACTION CONTROL LANGUAGE

1. **Commit the current transaction to make all changes permanent and end the transaction:**
   SQL>COMMIT;

   COMMIT COMPLETE.

2. **Commit a specific savepoint within a transaction and continue with the transaction:**

   SQL>COMMIT TO savepoint_abinaya;

   SAVEPOINT COMMITED.

3. **Commit the transaction and immediately start a new one:**
   SQL>COMMIT AND CHAIN;
   TRANSACTION COMMITED.

4. **Rollback the entire transaction, undoing all changes, and end the transaction:**

   SQL>ROLLBACK;

   ROLLBACK COMPLETED.

5. **Rollback to a specific savepoint within a transaction and continue with the transaction:**

   SQL>ROLLBACK TO savepoint_raj;

   ROLLBACK TO SAVEPOINT COMPLETED.

6. **Rollback to the start of the transaction, undoing all changes, and end the transaction:**
   SQL>ROLLBACK TO START;

7. **Set a savepoint within a transaction for a specific point:**
   SQL>SAVEPOINT savepoint_anukeerthana;
   SAVEPOINT VICKY ESTABLISHED.

8. **Set a savepoint within a transaction and specify a name:**
   SQL>SAVEPOINT custom_savepoint;

9. **Release a specific savepoint within a transaction:**
   SQL>RELEASE savepoint_anugrahaa;
   SAVEPOINT RELEASED.

10. **Release a custom savepoint within a transaction:**
    SQL>RELEASE custom_savepoint;

11. Start a new transaction explicitly:

SQL>BEGIN;

12 . Start a new transaction with a custom name:

SQL>BEGIN WORK;

13 . End the current transaction without making any changes permanent:

SQL>ROLLBACK WORK;

TRANSACTION ROLLED BACK

14 . **End the current transaction with a savepoint and continue with the next transaction:**

SQL>RELEASE savepoint_lia AND CHAIN;

15. **Commit the current transaction, making all changes permanent,and immedia**

immediately start a new one:

SQL>COMMIT AND BEGIN;

DATA CONTROL LANNGUAGE

1. SQL> grant select on customer to lia;

grant succeeded.

2. SQL> revoke select on customer from anugrahaa;

revoke succeeded.

3. SQL> grant insert,update on customer to anubarathi;

privileges granted.

4. SQL> revoke delete on customer from anukeerthana;

privileges revoked.

5. SQL> grant excute on function order_date to customer_id;

privileges granted.

6. SQL> grant usage on return_date to raj;

privileges granted.

7. SQL> revoke select on customer from abinaya;

privileges revoked.

8. SQL> revoke execute on procedure product_ordered from the customer_id101;

privileges revoked.

9. SQL> grant create temporary table to shalu;

privileges granted.

10. SQL> revoke connect from lia;

privileges revoked.

DATA QUERY LANGUAGE

1. **Retrive all customer names and their respective phone_number:**

SQL> select name,phone_num from customer;

NAME	PHONE_NUM
abinaya	9080766775
amirthavarshini	7668889899
anubarathi	9080318180
anugrahaa	8223566789
anukeerthana	7678887437
lia	8334679090
raj	7675435465
vicky	9614070476

2. **Retrive the details of customers who has cart_id=5:**

SQL> select customer_id,name,cart_id from customer where customer_id=105;

CUSTOMER_ID	NAME	CART_ID
105	anukeerthana	5

3. **Retrive the return amount by each customer on 15-feb-2023:**

SQL> Select return_id, SUM(return_amount) from return where return_date = TO_DATE('15-Feb-23', 'DD-MON-YY') group by return_id;

RETURN_ID	SUM(RETURN_AMOUNT)
506	3000

4. **Retrieve the payment type used by the customer with a total amount>500:**

```
SQL> select pm.payment_type from payment pm join customer c on
pm.customer_id=c.customer_id where pm.total_amount>500;
PAYMENT_TY
----------
upi
netbanking
gpay
online
cod
card
online
cod
```

5. **Retrive the name of the customer in the customer table:**

```
SQL> select name from customer where customer_id=102;
NAME
------------------------------
amirthavarshini
```

AGGREGATE FUNCTIONS AND SORTING

1. SQL>select avg(total_amount)as "average total_amount" from payment;

```
AVERAGE TOTAL_AMOUNT
----------------------------------------
        2750
```

2. SQL>select min(total_amount)as "minimum total_amount" from payment;

```
MINIMUM TOTAL_AMOUNT
--------------------
        1000
```

3. SQL>select max(total_amount)as "maximum total_amount" from payment;

```
MAXIMUM TOTAL_AMOUNT
--------------------
        4500
```

4. SQL> select count(customer_id)as "number of customers" from customer;

```
NUMBER OF CUSTOMERS
-------------------
          8
```

5. SQL>select sum(total_amount)as "total amount" from payment;

TOTAL AMOUNT

22000

6. SQL>select abs(20) as "absolute value" from dual;

ABSOLUTE VALUE

20

7. SQL>select round(1738.56) as “round” from dual;

ROUND

1739

8. SQL>select power(3,2) as “power” from dual;

POWER

9

9. SQL>select sqrt(25) as “square root” from dual;

SQUARE ROOT

5

10. SQL>select exp(5) as “exponent” from dual;

EXPONENT

148.413159

11. SQL>select extract(month from sysdate) as “month” from dual;

MONTH

9

12. SQL>select extract(year from date’2018-07-07’) as “year” from dual;

YEAR

2018

13. sql>select greatest(4,10,20) as “number” from dual;

NUMBER

20

14. SQL> select least(4,10,20) as "number" from dual;

NUMBER

4

15. SQL>select mod(15,8) as “number” from dual;

```
NUMBER
---------
        7
```

16. SQL> select trunc(138.356,1) as “number” from dual;

```
NUMBER
----------
     138.3
```

17. SQL> select ceil(38.6) as "number" from dual;

```
NUMBER
----------
        39
```

18. SQL> select to_date('4-jan-2023', 'dd-mon-yyyy') from dual;

```
TO_DATE
---------
04-JAN-23
```

19. SQL> select ltrim('customer_id') as “modifiedname” from customer;

```
ModifiedNam
-----------
customer_id
customer_id
customer_id
customer_id
customer_id
customer_id
customer_id
customer_id
```

20. SQL> select substr('welcome', 3, 2) from dual;

```
SU
--
LC
```

21. SQL> select ascii('a') from dual;

```
ASCII('A')
----------
        65
```

22. SQL> select lower(name) from customer;

```
LOWER(NAME)
------------------------------
  abinaya
```

amirthavarshini
anubarathi
anugrahaa
anukeerthana
raj
vicky
lia

23. SQL> select initcap(name) from customer;

INITCAP(NAME)

Abinaya
Amirthavarshini
Anubarathi
Anugrahaa
Anukeerthana
Raj
Vicky
Lia

6 rows selected.

24. SQL> select length('gpay') from dual;

LENGTH('GPAY')

4

25. SQL> select upper(name) from customer;

UPPER(NAME)

ABINAYA
AMIRTHAVARSHINI
ANUBARATHI
ANUGRAHAA
ANUKEERTHANA
RAJ
VICKY
LIA

7 rows selected.

26. SQL> select payment_type, sum(total_amount) as "total_amount" from payment group by payment_type;

PAYMENT_TY	TOTAL_AMOUNT
upi	1000
netbanking	1500
gpay	2000

online	6500
cod	7500
card	3500

6 rows selected.

27. SQL> select name from customer where instr(name, 'a') > 0;

NAME
---
abinaya
amirthavarshini
anubarathi
anugrahaa
anukeerthana
raj
lia

7 rows selected.

28. SQL> select payment_id, count(*) as "num_customers_payment"from payment group by payment_id;

PAYMENT_ID	NUM_CUSTOMERS_PAYMENT
201	1
202	1
203	1
204	1
205	1
206	1
207	1
208	1

8 rows selected.

29. SQL> select length('paxton') from dual;

LENGTH('PAXTON')

6

30. SQL> select avg(return_amount) as “average return_amount” from return;

AVERAGE RETURN_AMOUNT

2750

31. SQL> select payment_type, avg(total_amount) as "average total_amount" from payment group by payment_type;

PAYMENT_TY	AVERAGE TOTAL_AMOUNT
upi	1000
netbanking	1500
gpay	2000

online	3250
cod	3750
card	3500

6 rows selected.

32. SQL> select name from customer where instr(name, 'a') > 0;

NAME
---
abinaya
amirthavarshini
anubarathi
anugrahaa
anukeerthana
raj
lia

7 rows selected.

33. SQL> select avg(return_amount) as "average return_amount" from return;

AVERAGE RETURN_AMOUNT

2750

34. SQL> select sum(return_amount) as "total return amount" from return;

TOTAL RETURN AMOUNT

22000

35. SQL> select max(return_amount) as "highest return amount" from return;

HIGHEST RETURN AMOUNT

4500

JOINS

1. SQL>select customer.customer_id,name,payment.total_amount from customer inner join payment on customer.customer_id=payment.customer_id;

CUSTOMER_ID	NAME	TOTAL_AMOUNT
101	abinaya	1000
102	amirthavarshini	1500
103	anubarathi	2000
104	anugrahaa	2500
105	anukeerthana	3000

106	lia	3500
107	raj	4000
108	vicky	4500

2. SQL> select customer.customer_id,name,payment.payment_type from customer left join payment on customer.customer_id=payment.customer_id;

CUSTOMER_ID	NAME	PAYMENT_TY
101	abinaya	upi
102	amirthavarshini	netbanking
103	anubarathi	gpay
104	anugrahaa	online
105	anukeerthana	cod
106	lia	card
107	raj	online
108	vicky	cod

3. SQL> select customer.customer_id,name,payment.total_amount from customer right join payment on customer.customer_id=payment.customer_id;

CUSTOMER_ID	NAME	TOTAL_AMOUNT
101	abinaya	1000
102	amirthavarshini	1500
103	anubarathi	2000
104	anugrahaa	2500
105	anukeerthana	3000
106	lia	3500
107	raj	4000
108	vicky	4500

4. SQL>select customer.customer_id, customer.name, payment.payment_type from customer full outer join payment on customer.customer_id = payment.customer_id;

CUSTOMER_ID	NAME	PAYMENT_TY
101	abinaya	upi
102	amirthavarshini	netbanking
103	anubarathi	gpay
104	anugrahaa	online
105	anukeerthana	cod
106	lia	card
107	raj	online
108	vicky	cod

5. SQL> select customer.customer_id, customer.name, payment.total_amount from customer cross join payment;

CUSTOMER_ID	NAME	TOTAL_AMOUNT
101	abinaya	1000
102	amirthavarshini	1000
103	anubarathi	1000
104	anugrahaa	1000
105	anukeerthana	1000
106	lia	1000
107	raj	1000
108	vicky	1000
101	abinaya	1500
102	amirthavarshini	1500
103	anubarathi	1500
104	anugrahaa	1500
105	anukeerthana	1500
106	lia	1500
107	raj	1500
108	vicky	1500
101	abinaya	2000
102	amirthavarshini	2000
103	anubarathi	2000
104	anugrahaa	2000
105	anukeerthana	2000
106	lia	2000
107	raj	2000
108	vicky	2000
101	abinaya	2500
102	amirthavarshini	2500
103	anubarathi	2500

104	anugrahaa	2500
105	anukeerthana	2500
106	lia	2500
107	raj	2500
108	vicky	2500
101	abinaya	3000
102	amirthavarshini	3000
103	anubarathi	3000
104	anugrahaa	3000
105	anukeerthana	3000
106	lia	3000
107	raj	3000
108	vicky	3000
101	abinaya	3500
102	amirthavarshini	3500
103	anubarathi	3500
104	anugrahaa	3500
105	anukeerthana	3500
106	lia	3500
107	raj	3500
108	vicky	3500
101	abinaya	4000
102	amirthavarshini	4000
103	anubarathi	4000
104	anugrahaa	4000
105	anukeerthana	4000
106	lia	4000
107	raj	4000
108	vicky	4000
101	abinaya	4500

102	amirthavarshini	4500
103	anubarathi	4500
104	anugrahaa	4500
105	anukeerthana	4500
106	lia	4500
107	raj	4500
108	vicky	4500

SET OPERATIONS

1. SQL> select * from payment where total_amount<3000;

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
201	upi	1000	101	1
202	netbanking	1500	102	2
203	gpay	2000	103	3
204	online	2500	104	4

2. SQL> select * from payment where total_amount<=3000;

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
201	upi	1000	101	1
202	netbanking	1500	102	2
203	gpay	2000	103	3
204	online	2500	104	4
205	cod	3000	105	5

3. SQL> select * from payment where total_amount>=4000;

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
207	online	4000	107	7
208	cod	4500	108	8

4. SQL> select customer_id,payment_type,total_amount from payment where total_amount between 2000 and 4500;

CUSTOMER_ID	PAYMENT_TY	TOTAL_AMOUNT

103	gpay	2000
104	online	2500
105	cod	3000
106	card	3500
107	online	4000
108	cod	4500

6 rows selected.

5. SQL> select customer_id,payment_type,total_amount from payment where total_amount in(2000,3000,4000);

CUSTOMER_ID	PAYMENT_TY	TOTAL_AMOUNT
103	gpay	2000
105	cod	3000
107	online	4000

6. SQL> select customer_id,payment_type,total_amount from payment where total_amount not in(4500,2500,4000);

CUSTOMER_ID	PAYMENT_TY	TOTAL_AMOUNT
101	upi	1000
102	netbanking	1500
103	gpay	2000
105	cod	3000
106	card	3500

7. SQL> select customer_id,total_amount from payment where customer_id=102 and total_amount=1500;

CUSTOMER_ID	TOTAL_AMOUNT
102	1500

8. SQL> select customer_id,total_amount,(total_amount-500) from payment;

CUSTOMER_ID	TOTAL_AMOUNT	(TOTAL_AMOUNT-500)
101	1000	500
102	1500	1000
103	2000	1500
104	2500	2000
105	3000	2500
106	3500	3000
107	4000	3500
108	4500	4000

6 rows selected.

9. SQL> select customer_id from delivery intersect select customer_id from payment;

```
CUSTOMER_ID
-----------
       101
       102
       103
       104
       105
       106
       107
       108
```

8 rows selected.

10. SQL> select customer_id,total_amount from payment where total_amount>(select avg(total_amount)from payment);

```
CUSTOMER_ID TOTAL_AMOUNT
------------------  ----------------------
       105                  3000
       106                  3500
       107                  4000
       108                  4500
```

11. SQL> select customer_id,total_amount from payment where total_amount=(select min(total_amount) from payment);

```
CUSTOMER_ID  TOTAL_AMOUNT
-----------  ------------
       101         1000
```

12. SQL> select name,customer_id from customer where customer_id=102;

```
NAME                          CUSTOMER_ID
------------------------------ -----------
amirthavarshni                         102
```

13. SQL> select customer_id,total_amount from payment where total_amount=(select max(total_amount) from payment);

```
CUSTOMER_ID    TOTAL_AMOUNT
--------------------  ----------------------
       108                    4500
```

14. SQL> select customer_id,total_amount from payment where total_amount<3000 or total_amount>4000;

```
CUSTOMER_ID    TOTAL_AMOUNT
--------------------  ----------------------
       101         1000
```

102	1500
103	2000
104	2500
108	4500

15. SQL> select customer_id,total_amount from payment where payment_type='gpay' and total_amount>1000;

CUSTOMER_ID	TOTAL_AMOUNT
103	2000

16. SQL> select name,order_date from customer where order_date<to_date('04-jan-2023','dd-mm-yyyy');

NAME	ORDER_DAT
abinaya	01-JAN-23
amirthavarshni	02-JAN-23
anubarathi	03-JAN-23

VIEWS

1. SQL> create view customer_and_product as select customer.customer_id, product.product_id,product.type from customer inner join product on customer.customer_id=product.customer_id where customer.name='abinaya';

view created.

2. SQL> create view customernames_andph_ as select name,phone_num from customer where customer_id=102;

View created.

3. SQL> create view hightotalamount as select payment_id,total_amount from payment where total_amount>4000;

view created.

4. SQL> create view product_details_male as select product_id,type,colour,gender from product where gender='male';

view created.

5. SQL> create view payment_online as select payment_id,payment_type,total_amount from payment where payment_type='online';

view created.

6. SQL> create view product_quantity as select product_id,type,quantity from product;

view created.

7. SQL> create view return_details as select return_id,product_id,return_amount from return;

view created.

8. SQL> create view payment_netbank as select payment_id, payment_type, total_amount, cart_id from payment where payment_type='netbanking';

view created.

9. SQL> create view product_details_female as select product_id,type,colour,gender from product where gender='female';

view created.

10. SQL> create view delivery_address as select delivery_id,customer_id,address,pincode from delivery;

view created.

TABLES AND RECORDS

1. Display table structure(columns)for customer table:

SQL>desc customer;

Name	Null?	Type
CUSTOMER_ID	NOT NULL	NUMBER(10)
NAME		VARCHAR2(30)
PHONE_NUM		NUMBER(10)
ORDER_DATE		DATE
CART_ID		NUMBER(10)

2. Display first 4 records from the customer table:

SQL> select*from customer where rownum<=4;

CUSTOMER_ID	NAME	PHONE_NUM	ORDER_DAT	CART_ID
101	abinaya	9080766775	01-JAN-23	1
102	amirthavarshini	7668889899	02-JAN-23	2
103	anubarathi	9080318180	03-JAN-23	3
104	anugrahaa	8223566789	04-JAN-23	4

3. Display payment structure(columns)for payment table:

SQL> desc payment;

Name	Null?	Type
PAYMENT_ID	NOT NULL	NUMBER(10)
PAYMENT_TYPE		VARCHAR2(10)
TOTAL_AMOUNT		NUMBER(10)
CUSTOMER_ID		NUMBER
CART_ID		NUMBER(10)

4. Display first 4 records from the payment table:

SQL> select*from payment where rownum<=4;

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
201	upi	1000	101	1
202	netbanking	1500	102	2
203	gpay	2000	103	3
204	online	2500	104	4

5. Display table structure(columns)for delivery table:

SQL> desc delivery;

Name	Null?	Type
DELIVERY_ID	NOT NULL	NUMBER
CUSTOMER_ID		NUMBER
ADDRESS		VARCHAR2(30)
PINCODE		NUMBER(6)
PHONE_NUM		NUMBER(10)
DELIVERY_DATE		DATE

6. Display first 7 records from the delivery table:

SQL> select delivery_id,address from delivery where rownum<=7;

DELIVERY_ID	ADDRESS
402	chennai
403	madurai
404	trichy
405	kerala
406	pollachi
407	erode
408	andhrapradesh

7. Display all unique customer IDs from the customer table:

SQL> select distinct customer_id from customer;

CUSTOMER_ID
101
102
103
104
105

106

107

108

8. Find the customer with max phone_num:

SQL> select name,phone_num from customer where phone_num=(select max(phone_n um)from customer);

NAME	PHONE_NUM
vicky	9614070476

9. List employee with cart_id='03':

SQL> select name,order_date from customer where cart_id=03;

NAME	ORDER_DAT
anubarathi	03-JAN-23

10.To retrive information of customer table:

```
SQL> DECLARE
  2  customer_rec customer%ROWTYPE;
  3  BEGIN
  4  SELECT * INTO customer_rec FROM customer WHERE CUSTOMER_ID = 104;
  5  DBMS_OUTPUT.PUT_LINE('CUSTOMER ID: ' || customer_rec.customer_id);
  6  DBMS_OUTPUT.PUT_LINE('NAME: ' || customer_rec.name);
  7  DBMS_OUTPUT.PUT_LINE('PHONE_NUM: ' || customer_rec.phone_num);
  8  DBMS_OUTPUT.PUT_LINE('ORDER_DATE: ' || customer_rec.order_date);
  9  DBMS_OUTPUT.PUT_LINE('CART_ID: ' || customer_rec.cart_id);
 10  END;
 11  /

CUSTOMER ID: 104
```

NAME: anugrahaa

PHONE_NUM: 8223566789

ORDER_DATE: 04-JAN-23

CART_ID: 4

PL/SQL procedure successfully completed.

PL/SQL QUERIES

1. SQL> DECLARE

2 totalAmount NUMBER(10, 2);

3 customerId INT;

4 paymentType VARCHAR2(20);

5 BEGIN

6 FOR payment_rec IN (SELECT CUSTOMER_ID, PAYMENT_TYPE, TOTAL_AMOUNT FROM payment) LOOP

7 customerId := payment_rec.CUSTOMER_ID;

8 paymentType := payment_rec.PAYMENT_TYPE; -- Use PAYMENT_TYPE

9 totalAmount := payment_rec.TOTAL_AMOUNT;

10 -- Output in a single line

11 DBMS_OUTPUT.PUT_LINE('Customer ' || customerId || ' made a payment of ' || totalAmount || ' using ' || paymentType || '.');

12 END LOOP;

13 END;

14 /

Customer 101 made a payment of 1000 using upi.

Customer 102 made a payment of 1500 using netbanking.

Customer 103 made a payment of 2000 using gpay.

Customer 104 made a payment of 2500 using online.

Customer 105 made a payment of 3000 using cod.

Customer 106 made a payment of 3500 using card.

Customer 107 made a payment of 4000 using online.

Customer 108 made a payment of 4500 using cod.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

2. SQL> DECLARE

```
  2  CURSOR delivery_cursor IS
  3  SELECT DELIVERY_ID, CUSTOMER_ID, ADDRESS, PINCODE, PHONE_NUM
  4  FROM delivery;  -- Make sure the table name matches your database
  5  BEGIN
  6  FOR delivery_rec IN delivery_cursor LOOP
  7  DBMS_OUTPUT.PUT_LINE('Delivery ID: ' || delivery_rec.DELIVERY_ID ||
  8  ', Customer ID: ' || delivery_rec.CUSTOMER_ID ||
  9  ', Address: ' || delivery_rec.ADDRESS ||
 10  ', Pincode: ' || delivery_rec.PINCODE ||
 11  ', Phone Number: ' || delivery_rec.PHONE_NUM);
 12  END LOOP;
 13  END;
 14  /
```

Delivery ID: 401, Customer ID: 101, Address: coimbatore, Pincode: 653234, Phone Number: 908765778

Delivery ID: 402, Customer ID: 102, Address: chennai, Pincode: 625987, Phone Number: 6547893426

Delivery ID: 403, Customer ID: 103, Address: madurai, Pincode: 654378, Phone Number: 9087656334

Delivery ID: 404, Customer ID: 104, Address: trichy, Pincode: 600987, Phone Number: 7865786543

Delivery ID: 405, Customer ID: 105, Address: kerala, Pincode: 656578, Phone

Number: 765892345

Delivery ID: 406, Customer ID: 106, Address: pollachi, Pincode: 645789, Phone

Number: 7675435465

Delivery ID: 407, Customer ID: 107, Address: erode, Pincode: 651342, Phone

Number: 8334679090

Delivery ID: 408, Customer ID: 108, Address: andhrapradesh, Pincode: 675876,

Phone Number: 9614070476

PL/SQL procedure successfully completed.

PACKAGE

```
1. SQL> Create Or Replace Package Body Customer_Package As
   2  Procedure Display_Customers Is
   3  Cursor Customer_Cursor Is
   4  Select Customer_Id, Name, Phone_Num, Order_Date, Cart_Id  -- Adjusted Column
Name
   5  From Customer;  -- Ensure The Table Name Is Correct
   6  Begin
   7  For Customer_Rec In Customer_Cursor Loop
   8  Dbms_Output.Put_Line('Customer Id: ' || Customer_Rec.Customer_Id ||
   9  ', Name: ' || Customer_Rec.Name ||
  10   ', Phone: ' || Customer_Rec.Phone_Num ||
  11   ', Order Date: ' || Customer_Rec.Order_Date ||  -- Adjusted Column Name
  12  ', Cart Id: ' || Customer_Rec.Cart_Id);
  13   End Loop;
  14   End Display_Customers;
  15  End Customer_Package;
  16  /
```

Package Body Created.

```
SQL> Create Or Replace Package Body Customer_Package As
```

```
2      Procedure Display_Customers Is
3  Cursor Customer_Cursor Is
4  Select Customer_Id, Name, Phone_Num, Order_Date, Cart_Id  -- Use The Correct
Column Name
5  From Customer;
6   Begin
7  For Customer_Rec In Customer_Cursor Loop
8  Dbms_Output.Put_Line('Customer Id: ' || Customer_Rec.Customer_Id ||
9  ', Name: ' || Customer_Rec.Name ||
10  ', Phone: ' || Customer_Rec.Phone_Num ||
11  ', Order Date: ' || Customer_Rec.Order_Date ||
12  ', Cart Id: ' || Customer_Rec.Cart_Id);
13  End Loop;
14  End Display_Customers;
15  End Customer_Package;
16  /
```

Package Body Created.

```
SQL> Show Errors Package Body Customer_Package;
No Errors.
Sql> Set Serveroutput On;
Sql>
Sql> Begin
2   Customer_Package.Display_Customers;
3  End;
4  /
```

Customer Id: 101, Name: Abinaya, Phone: 9080766775, Order Date: 01-Jan-23, Cart Id: 1

Customer Id: 102, Name: Amirthavarshini, Phone: 7668889899, Order Date: 02-Jan-23, Cart Id: 2

Customer Id: 103, Name: Anubarathi, Phone: 9080318180, Order Date: 03-Jan-23,

Cart Id: 3

Customer Id: 104, Name: Anugrahaa, Phone: 8223566789, Order Date: 04-Jan-23,

Cart Id: 4

Customer Id: 105, Name: Anukeerthana, Phone: 7678887437, Order Date: 05-Jan-23,

Cart Id: 5

Customer Id: 106, Name: Raj, Phone: 7675435465, Order Date: 06-Jan-23, Cart Id:

6

Customer Id: 107, Name: Vicky, Phone: 8334679090, Order Date: 07-Jan-23, Cart

Id: 7

Customer Id: 108, Name: Lia, Phone: 9614070476, Order Date: 08-Jan-23, Cart Id:

8

Pl/Sql Procedure Successfully Completed.

TRIGGER

```
1. SQL> CREATE TABLE product_log (
  2     log_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
  3     product_id NUMBER,
  4     change_type VARCHAR2(10),
  5     change_date DATE,
  6     PRIMARY KEY (log_id)
  7  );
Table created.

SQL> CREATE OR REPLACE TRIGGER log_product_changes
2  AFTER INSERT OR UPDATE ON product
3  FOR EACH ROW
4  BEGIN
5  IF INSERTING THEN
6  INSERT INTO product_log (product_id, change_type, change_date)
```

```
 7  VALUES (:NEW.PRODUCT_ID, 'INSERT', SYSDATE);
 8   ELSIF UPDATING THEN
 9   INSERT INTO product_log (product_id, change_type, change_date)
10       VALUES (:NEW.PRODUCT_ID, 'UPDATE', SYSDATE);
11  END IF;
12  END;
 13  /
```

Trigger created.

SQL> INSERT INTO product (PRODUCT_ID, CUSTOMER_ID, TYPE, COLOUR, GENDER, QUANTITY) VALUES (309, 109, 'casual', 'purple', 'female', 3);

SQL> UPDATE product SET QUANTITY = 5 WHERE PRODUCT_ID = 301;

1 row updated.

SQL> SELECT * FROM product_log;

LOG_ID	PRODUCT_ID	CHANGE_TYP	CHANGE_DA
2	301	UPDATE	18-SEP-24

2. SQL> ALTER TABLE PAYMENT ADD (PAYMENT_AMOUNT NUMBER, PAYMENT_DATE DATE);

Table altered.

```
SQL> CREATE OR REPLACE TRIGGER AutomaticPaymentTrigger
 2  BEFORE INSERT ON PAYMENT
 3  FOR EACH ROW
 4  BEGIN
 5     -- Automatically set PAYMENT_AMOUNT based on TOTAL_AMOUNT
 6     :NEW.PAYMENT_AMOUNT := :NEW.TOTAL_AMOUNT;
 7    -- Set the payment date to the current date
 8     :NEW.PAYMENT_DATE := SYSDATE;
```

```
 9  END;
10  /
```

Trigger created.

3. SQL> CREATE TABLE SALES (

```
 2   SALE_ID NUMBER PRIMARY KEY,
 3   PRODUCT_ID NUMBER,
 4   SALE_QUANTITY NUMBER,
 5   SALE_DATE DATE
 6  );
```

Table created.

SQL> INSERT INTO SALES (SALE_ID, PRODUCT_ID, SALE_QUANTITY, SALE_DATE)VALUES (1, 301, 3, SYSDATE);

1 row created.

SQL> CREATE OR REPLACE TRIGGER display_product_changes

```
 2  BEFORE UPDATE ON PRODUCT
 3  FOR EACH ROW
 4  DECLARE
 5     quantity_diff NUMBER;
 6  BEGIN
 7     -- Calculate the quantity difference
 8     quantity_diff := :NEW.QUANTITY - :OLD.QUANTITY;
 9     -- Output the old product details, new product details, and quantity difference
10      DBMS_OUTPUT.PUT_LINE('Old Product ID: ' || :OLD.PRODUCT_ID);
11      DBMS_OUTPUT.PUT_LINE('Old Customer ID: ' || :OLD.CUSTOMER_ID);
12      DBMS_OUTPUT.PUT_LINE('Old Type: ' || :OLD.TYPE);
13      DBMS_OUTPUT.PUT_LINE('Old Colour: ' || :OLD.COLOUR);
14      DBMS_OUTPUT.PUT_LINE('Old Gender: ' || :OLD.GENDER);
15      DBMS_OUTPUT.PUT_LINE('Old Quantity: ' || :OLD.QUANTITY);
16      DBMS_OUTPUT.PUT_LINE('New Product ID: ' || :NEW.PRODUCT_ID);
```

```
17      DBMS_OUTPUT.PUT_LINE('New Customer ID: ' || :NEW.CUSTOMER_ID);
18      DBMS_OUTPUT.PUT_LINE('New Type: ' || :NEW.TYPE);
19      DBMS_OUTPUT.PUT_LINE('New Colour: ' || :NEW.COLOUR);
20      DBMS_OUTPUT.PUT_LINE('New Gender: ' || :NEW.GENDER);
21      DBMS_OUTPUT.PUT_LINE('New Quantity: ' || :NEW.QUANTITY);
22      DBMS_OUTPUT.PUT_LINE('Quantity Difference: ' || quantity_diff);
23  END;
24  /
```

Trigger created.

```
SQL> UPDATE PRODUCT
  2  SET QUANTITY = 10
  3  WHERE PRODUCT_ID = 301;
```

Old Product ID: 301

Old Customer ID: 101

Old Type: western

Old Colour: orange

Old Gender: female

Old Quantity: 5

New Product ID: 301

New Customer ID: 101

New Type: western

New Colour: orange

New Gender: female

EXCEPTION HANDLING

1. Exception program to find the specific customer:

```
SQL> DECLARE
  2  v_customer_name VARCHAR2(100);
  3  BEGIN
```

```
  4  SELECT name
  5  INTO v_customer_name
  6  FROM customer
  7  WHERE customer_id=101;
  8  DBMS_OUTPUT.PUT_LINE('Customer Name: ' ||v_customer_name);
  9  EXCEPTION
 10  WHEN NO_DATA_FOUND THEN
 11  DBMS_OUTPUT.PUT_LINE('Customer not found.');
 12  WHEN OTHERS THEN
 13  DBMS_OUTPUT.PUT_LINE('An error occurred.');
 14  END;
 15  /
```

Customer Name: abinaya

PL/SQL procedure successfully completed.

2. Exception program to find the specific customer:

```
SQL> DECLARE
 2 v_customer_name VARCHAR2(100);
 3 BEGIN
 4 SELECT name
 5 INTO v_customer_name
 6 FROM customer
 7 WHERE customer_id =600;
 8 DBMS_OUTPUT.PUT_LINE('customer Name: ' || v_customer_name);
 9 EXCEPTION
10 WHEN NO_DATA_FOUND THEN
11 DBMS_OUTPUT.PUT_LINE('Customer not found.');
12 WHEN OTHERS THEN
13 DBMS_OUTPUT.PUT_LINE('An error occurred.');
14 END;
```

```
 15 /
```

Customer not found.

PL/SQL procedure successfully completed.

3. Exception Program To Delete A Particular Column:

```
SQL> DECLARE
 2 customer_id_input VARCHAR2(10):='3';
 3 BEGIN
 4 DELETE FROM customer
 5 WHERE customer_id=customer_id_input;
 6 DBMS_OUTPUT.PUT_LINE('customer deleted successfully');
 7 EXCEPTION
 8 WHEN OTHERS THEN
 9 DBMS_OUTPUT.PUT_LINE('cannot delete customer due to references in other
   tables');
10 END;
11 /
```

customer deleted successfully

PL/SQL procedure successfully completed.

SUB QUERIES

1. Add a new column named monthly_emi to the “payment” table:

```
SQL> alter table payment add monthly_emi number(5);
```

Table altered.

2. Modify the data type of the “payment_type” column to varchar:

```
SQL> alter table payment modify payment_type varchar(25);
```

Table altered.

3. Update the monthly_emi amount for an particular customer:

SQL> update payment set monthly_emi=500 where customer_id=108;

1 row updated.

4. Add a new column named “name” with varchar data type to the delivery table:

SQL> alter table delivery add name varchar(20);

Table altered.

5. Rename the payment table to payment_details:

SQL> alter table payment rename to payment_details;

Table altered.

6. Update the payment_type for a specific id(payment_id):

SQL> update payment set payment_type = 'gpay' where payment_id=204;

1 row updated.

7. Query to get the total_amount with the payment type from the payment table:

SQL> select payment_type, sum(total_amount) as total_amount from payment group by payment_type;

PAYMENT_TY	TOTAL_AMOUNT
upi	1000
netbanking	1500
gpay	2000
online	6500
cod	7500
card	3500

8. Query to get the delivery_id,customer_id, address, pincode, phone_num from the delivery table where the address is equal to Coimbatore:

SQL> select delivery_id, customer_id, address, pincode, phone_num from delivery where address = 'coimbatore';

DELIVERY_ID	CUSTOMER_ID	ADDRESS	PINCODE	PHONE_NUM
401	101	coimbatore	653234	908765778

9. Query to select the maximum of total_amount from the payment table:

SQL> select payment_id, payment_type, total_amount from payment where total_amount=(select max(total_amount)from payment);

PAYMENT_ID	PAYMENT_TYPE	TOTAL_AMOUNT
208	cod	4500

10. Query to get the customer name where the total_amount(>3000):

SQL> select name from customer where customer_id in(select customer_id from payment where total_amount>3000);

NAME

---

lia

raj

vicky

11. Query to get the customer name where payment type=cod:

SQL> select name from customer where customer_id in(select customer_id from payment where payment_type='cod');

NAME

---

anukeerthana

Vicky

12. Query to get the total_amount from payment table where payment type = online:

SQL> select sum(total_amount) from payment where payment_type='online';

SUM(TOTAL_AMOUNT)

4000

13. Query to get the name where total amount <4000:

SQL> select name from customer where customer_id in(select customer_id from payment where total_amount<4000);

NAME

---

abinaya

amirthavarshni

anubarathi

anugrahaa

anukeerthana

lia

6 rows selected.

14. Display the customer name where the payment types are not in “cod, card”:

SQL> select name from customer where customer_id in(select customer_id from payment where payment_type not in('cod','card'));

NAME

---

abinaya

amirthavarshni

anubarathi

anugrahaa

raj

15. SQL> select payment_id,payment_type,total_amount from payment where total_amount>(select avg(total_amount)from payment);

PAYMENT_ID	PAYMENT_TYPE	TOTAL_AMOUNT
205	cod	3000
206	card	3500
207	online	4000
208	cod	4500

16. Retrieve the customers with an even numbered customer's id:

SQL> select * from customer where mod(customer_id,2)=0;

CUSTOMER_ID	NAME	PHONE_NO	ORDER_DAT	CART_ID
102	amirthavarshni	7668889899	02-JAN-23	2
104	anugrahaa	8223566789	04-JAN-23	4
106	lia	8334679090	06-JAN-23	6
108	vicky	9614070476	08-JAN-23	8

17. To find the customers who does not have the total amount>2000:

SQL> select customer_id,name from customer where customer_id not in(select customer_id from payment where total_amount>2000);

CUSTOMER_ID	NAME
101	abinaya
103	anubarathi
102	amirthavarshni

18. SQL> create table product_promotions as

2 select customer_id,payment_type,total_amount,

3 case

4 when total_amount>4000 then 'high'

5 when total_amount>2000 then 'medium'

6 else'average'

```
  7  end as promotion
  8  from payment;
```

Table created.

19. To drop the table named “product_promotion”:

```
SQL> drop table product_promotions;
```

Table dropped.

20. To find the customer who are paying the amount through online:

```
SQL> select payment_id, payment_type, total_amount, customer_id, cart_id,
payment_amount from payment where payment_type = 'online';
```

PAYMENT_ID	PAYMENT_TY	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID	PAYMENT_AMOUNT
204	online	2500	104	4	
207	online	4000	107	7	

21. SQL> select delivery_id, customer_id, address, pincode, phone_num from delivery where delivery_date > to_date('01-feb-2023', 'dd-mon-yyyy');

DELIVERY_ID	CUSTOMER_ID	ADDRESS	PINCODE	PHONE_NUM
402	102	chennai	625987	6547893426
403	103	madurai	654378	9087656334
404	104	trichy	600987	7865786543
405	105	kerala	656578	765892345
406	106	pollachi	645789	7675435465
407	107	erode	651342	8334679090
408	108	andhrapradesh	675876	9614070476

22. To truncate all data in the “return” table to remove all the return records:

SQL> truncate table return;

Table truncated.

23. To find the customer who chose the product type should be traditional:

SQL> select product_id, customer_id, type, colour, gender, quantity from product where type = 'traditional';

PRODUCT_ID	CUSTOMER_ID	TYPE	COLOUR	GENDER	QUANTITY
302	102	traditional	red	male	2
306	106	traditional	white	female	2

24. To find the delivery date where the delivery date is equal to (01-feb-2023):

SQL> select delivery_id, customer_id, address, pincode, phone_num from delivery where delivery_date = to_date('01-feb-2023', 'dd-mon-yyyy');

DELIVERY_ID	CUSTOMER_ID	ADDRESS	PINCODE	PHONE_NUM
401	101	coimbatore	653234	908765778

25. To find the customer who use “online” payment (payment_type):

SQL> select customer_id,payment_id, payment_type from payment where payment _type='online';

CUSTOMER_ID	PAYMENT_ID	PAYMENT_TYPE
107	207	online

26. Retrieve the customer’s name and order_date:

SQL> select name,order_date from customer;

NAME	ORDER_DAT
abinaya	01-JAN-23
amirthavarshni	02-JAN-23

anubarathi	03-JAN-23
anugrahaa	04-JAN-23
anukeerthana	05-JAN-23
lia	06-JAN-23
raj	07-JAN-23
vicky	08-JAN-23

8 rows selected.

27. Get the details of the customer for a specific department id:

SQL> select * from customer where cart_id=2;

CUSTOMER_ID	NAME	PHONE_NO	ORDER_DAT	CART_ID
102	amirthavarshni	7668889899	02-JAN-23	2

28. Calculate the average total amount of the orders:

SQL> select avg(total_amount) as average_total_amount from payment;

AVERAGE_TOTAL_AMOUNT
2750

29. To find the return amount where the return amount should be greater than 2000:

SQL> select return_id, customer_id, product_id, return_date, return_amount from return where return_amount > 2000;

RETURN_ID	CUSTOMER_ID	PRODUCT_ID	RETURN_DA	RETURN_AMOUNT
504	104	304	14-FEB-23	2500
505	105	305	15-FEB-23	3000
506	106	306	16-FEB-23	3500
507	107	307	17-FEB-23	4000
508	108	308	18-FEB-23	4500

30. To find the average salary for payment_type:

SQL> select payment_type,avg(total_amount) as average_total_amount from payment group by payment_type;

PAYMENT_TYPE	AVERAGE_TOTAL_AMOUNT
upi	1000
netbanking	1500
gpay	2250
cod	3750
card	3500
online	4000

6 rows selected.

31. Find the customers who were ordered before 04-jan-2023:

SQL> select name,order_date from customer where to_date(order_date,'dd-mm-yy')<to_date('04-jan-23','dd-mm-yy');

NAME	ORDER_DAT
abinaya	01-JAN-23
amirthavarshni	02-JAN-23
anubarathi	03-JAN-23

32. Find the sum of total amount from the payment:

SQL> select sum(total_amount) as total_payments from payment;

TOTAL_PAYMENTS
22000

33. Retrieve the earliest ordered date among all:

SQL> select min(order_date)as earliest_date from customer;

EARLIEST_

01-JAN-23

34. List the customers who have a total_amount more than 4000:

SQL> select * from payment where total_amount>4000;

PAYMENT_ID	PAYMENT_TYPE	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
208	cod	4500	108	8

35. Find the sum of total amount for each customer id:

SQL> select customer_id,sum(total_amount) from payment group by customer_id;

CUSTOMER_ID	SUM(TOTAL_AMOUNT)
101	1000
102	1500
103	2000
104	2500
105	3000
106	3500
107	4000
108	4500

8 rows selected.

36. To find the specific details by using the return date:

SQL> select return_id, customer_id, product_id, return_date, return_amount from return where return_date = to_date('14-feb-2023', 'dd-mon-yyyy');

RETURN_ID	CUSTOMER_ID	PRODUCT_ID	RETURN_DA	RETURN_AMOUNT
504	104	304	14-FEB-23	2500

37. To find the customer's id with the highest total_amount:

SQL> select customer_id,total_amount from payment where total_amount=(select max(total_amount)from payment);

CUSTOMER_ID	TOTAL_AMOUNT
108	4500

38. To calculate(sum) the total_amount:

SQL> select sum(total_amount) from payment;

SUM (TOTAL_AMOUNT)
22000

39. To find the average value for the total_amount:

SQL> select avg(total_amount) from payment;

AVG(TOTAL_AMOUNT)
2750

40. Retrieve the specific customer details by using the customer id:

SQL> select name,order_date,phone_no from customer where customer_id=102;

NAME	ORDER_DAT	PHONE_NO
amirthavarshni	02-JAN-23	7668889899

41. Find the customer_id, payment_id, total_amount by using the payment_method:

SQL> select customer_id,payment_id,total_amount from payment where payment_ type='netbanking';

CUSTOMER_ID	PAYMENT_ID	TOTAL_AMOUNT
102	202	1500

42. Retrieve the customer details by using the address:

SQL> select customer_id,pincode,phone_num from delivery where address='chennai';

CUSTOMER_ID	PINCODE	PHONE_NUM
102	625987	6547893426

43. Retrieve the customer's id and names with phone_no starting with '7':

SQL> select customer_id,name,phone_no from customer where phone_no like'7%';

CUSTOMER_ID	NAME	PHONE_NO
102	amirthavarshni	7668889899
105	anukeerthana	7678887437
107	raj	7675435465

44. List all payments with payment_type containing 'cod':

SQL> select * from payment where payment_type like'cod%';

PAYMENT_ID	PAYMENT_TYPE	TOTAL_AMOUNT	CUSTOMER_ID	CART_ID
205	cod	3000	105	5
208	cod	4500	108	8

45. To find the customers name which are starting with the letter 'a':

SQL> select * from customer where name like'a%';

CUSTOMER_ID	NAME	PHONE_NO	ORDER_DAT	CART_ID
101	abinaya	9080766775	01-JAN-23	1
102	amirthavarshni	7668889899	02-JAN-23	2
103	anubarathi	9080318180	03-JAN-23	3

104	anugrahaa	8223566789	04-JAN-23	4
105	anukeerthana	7678887437	05-JAN-23	5

46. Retrieve online payment_method:

SQL> select customer_id,payment_id,payment_type,total_amount from payment w here payment_type='online';

CUSTOMER_ID	PAYMENT_ID	PAYMENT_TYPE	TOTAL_AMOUNT
107	207	online	4000

47.Retrieve specific customer details by using the cart_id:

SQL> select customer_id,name,phone_no,order_date from customer where cart_id=2;

CUSTOMER_ID	NAME	PHONE_NO	ORDER_DAT
102	amirthavarshni	7668889899	02-JAN-23

48. Retrieve the customer's id, payment type with a total_amount which is greater than 4000:

SQL> select customer_id,payment_type,total_amount from payment where total_ amount>4000;

CUSTOMER_ID	PAYMENT_TYPE	TOTAL_AMOUNT
108	cod	4500

49. Retrieve the customer details before 03-jan-23:

SQL> select customer_id,name,phone_no,order_date from customer where to_date (order_date,'dd-mm-yy')<to_date('03-jan-23','dd-mm-yy');

CUSTOMER_ID	NAME	PHONE_NO	ORDER_DAT
101	abinaya	9080766775	01-JAN-23

102	amirthavarshni	7668889899	02-JAN-23

50. To retrieve the customers with cart_id:

SQL> select customer_id,name,phone_no,cart_id from customer where cart_id i n(1,2);

CUSTOMER_ID	NAME	PHONE_NO	CART_ID
101	abinaya	9080766775	1
102	amirthavarshni	7668889899	2

51.Retrieve the customers with the cart_id (by applying not in):

SQL> select customer_id,name,phone_no,cart_id from customer where cart_id not in(1,2);

CUSTOMER_ID	NAME	PHONE_NO	CART_ID
103	anubarathi	9080318180	3
104	anugrahaa	8223566789	4
105	anukeerthana	7678887437	5
106	lia	8334679090	6
107	raj	7675435465	7
108	vicky	9614070476	8

6 rows selected.

CONCLUSION:

In conclusion, an eCommerce management system is essential for efficiently managing online shopping operations. It integrates various functionalities such as product catalog management, order processing, customer relationship management, and payment processing into a cohesive platform. By leveraging such a system, businesses can enhance operational efficiency, provide a seamless shopping experience, and effectively handle customer interactions. Key benefits include streamlined inventory management, improved sales tracking, and better customer insights, all of which contribute to increased customer satisfaction and business growth.

Product Information: Storing and managing product details, including descriptions, prices, and inventory levels, ensures that customers have accurate and up-to-date information.

Customer Data: Maintaining comprehensive customer profiles and order histories allows for personalized shopping experiences, targeted marketing, and enhanced customer service.

Order Management: Tracking orders from placement to delivery helps in managing fulfillment processes, handling returns, and ensuring customer satisfaction.

Payment Processing: Safeguarding and processing transactions securely integrates with financial systems to manage payments and refunds efficiently.